# Analisis Komparatif Metode Naif, Boids, dan RVO dalam Simulasi Kerumunan

## Optimisasi Divide and Conquer dengan QuadTree pada Boids dan RVO

Authors Rafael Marchel Darma Wijaya - 13523146
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: rafaelmarchel.dw@gmail.com , 13523146@std.stei.itb.ac.id

***Abstract*—This research presents a comparative analysis of three crowd simulation methods, Naive, Boids, and Reciprocal Velocity Obstacles (RVO), with a focus on collision avoidance optimization. Each agent in the simulation is modeled as an autonomous entity moving towards a specific goal in a circular scenario. The Naïve method is implemented as a baseline benchmark, while the Boids (reactive) and RVO (predictive) methods are optimized using a QuadTree data structure to enhance neighbor search efficiency. Experiments were conducted with 100 agents, and performance metrics evaluated include simulation duration, number of collisions, frames per second (FPS), computation time per frame, and average distance traveled. The results indicate that the Naïve method, while the fastest (6.62 seconds), produced the highest number of collisions (13,456). The Boids with QuadTree method significantly reduced collisions by 85% (1,926 collisions) but had the longest simulation duration (17.84 seconds). The RVO with QuadTree method proved to be the most effective, with only 126 collisions and the highest FPS (77.87), despite having the most complex computation time per frame (2.859 ms). This study concludes that RVO with QuadTree offers the best balance between simulation smoothness and collision avoidance effectiveness, while QuadTree optimization is proven essential for improving performance in complex algorithms. (*Abstract*)**

***Keywords—boids; divide and conquer; quadtree; RVO (key words)***

## I. Pendahuluan

Simulasi kerumunan telah menjadi bidang studi yang semakin penting dalam ilmu komputasi karena penerapannya yang luas dan berdampak langsung dalam berbagai sektor seperti perencanaan tata kota, manajemen evakuasi darurat, hingga industri hiburan digital [1]. Dalam konteks ini, pendekatan mikroskopis menjadi paradigma dominan. Tiap individu dalam kerumunan dimodelkan sebagai agen otonom yang berinteraksi berdasarkan aturan lokal. Pendekatan *bottom-up* ini memungkinkan terciptanya perilaku kolektif yang kompleks seperti formasi jalur yang alami, pembentukan antrian, atau bahkan kemacetan yang berasal dari interaksi lokal antar-agen [2].

Karya tulis ini bertujuan untuk mengevaluasi, membandingkan, dan mengoptimisasi tiga pendekatan fundamental dalam algoritma penghindaran tabrakan: (1) metode Naif tanpa aturan, (2) pendekatan Boids, yang berfokus pada aturan separasi, serta (3) metode Reciprocal Velocity Obstacles (RVO), yang menggunakan pendekatan prediktif berbasis kecepatan untuk menghindari tabrakan. Ketiga pendekatan ini memiliki kelebihan masing-masing.

Tujuan dari karya tulis ini adalah mengaplikasikan optimisasi algoritma dengan Divide and Conquer metode QuadTree agar algoritma-algoritma simulasi kerumunan dapat lebih efisien dan cepat. Diharapkan dengan peningkatan performa, algoritma simulasi kerumunan dapat dimodifikasi agar memiliki perilaku yang lebih kompleks agar menyerupai dunia nyata dan membantu memodelkan perilaku kerumunan makhluk hidup yang organik.

Struktur dari makalah ini disusun sebagai berikut: Bagian II menyajikan tinjauan pustaka dan landasan teori dari metode-metode yang dikaji. Bagian III membahas perancangan model kepanikan, integrasinya ke dalam algoritma yang diuji, serta metodologi eksperimen. Bagian IV menyajikan hasil dan analisis dari simulasi, yang kemudian diikuti oleh kesimpulan dan saran untuk penelitian selanjutnya di Bagian V.

## II. Dasar Teori dan Implementasi

Bagian ini akan menyajikan algoritma-algoritma penghindaran tabrakan yang menjadi dasar penelitian ini serta memberikan konteks mengenai upaya pemodelan psikologis dalam simulasi kerumunan.

### *A. Algoritma Penghindaran Tabrakan Mikroskopis*

Dalam model simulasi mikroskopis, atau dikenal juga sebagai model "bottom-up", setiap entitas direpresentasikan sebagai agen dengan properti seperti ukuran tubuh dan kecepatan berjalan [1]. Selain itu, setiap agen memiliki motivasi tertentu, seperti tujuan lokasi yang ingin dicapai. Pada setiap langkah simulasi, agen memperbarui kecepatannya dengan mempertimbangkan keberadaan agen lain di sekitarnya, sesuai dengan aturan tertentu yang mengatur

perilaku lokal mereka. Karya tulis ini akan mengeksplorasi tiga metode, metode naif yang menjadi ukuran dasar dan pembanding, metode boids dan RVO sebagai algoritma untuk agen pintar.

### *B. Metode Naif*

Metode naif merupakan pendekatan simulasi kerumunan tanpa adanya upaya untuk menghindari tabrakan. Setiap agen akan memiliki dorongan untuk pergi ke arah target. Namun, agen tidak akan memedulikan agen yang lain. Dengan kata lain, agen tidak akan memiliki pengaruh terhadap pergerakan atau sifat agen lainnya. Algoritma yang mengimplementasikan metode naif ini akan dinamakan Algoritma Ghost.

Algoritma Ghost mengimplementasikan strategi pergerakan yang paling naif, yaitu bergerak langsung menuju tujuan tanpa mempertimbangkan rintangan atau agen lain di sekitarnya. Mekanisme ini dapat dijelaskan sebagai berikut: (1) vektor arah dihitung dengan mengurangkan posisi agen saat ini dari posisi tujuan, (2) vektor arah dinormalisasi untuk mendapatkan vektor unit dengan panjang 1, (3) vektor unit dikalikan dengan kecepatan maksimum agen sehingga agen akan bergerak dengan kecepatan maksimum, (4) Kecepatan hasil perhitungan ditetapkan sebagai kecepatan agen. Ketika agen telah mencapai tujuan, kecepatan agen diatur menjadi nol sehingga agen berhenti bergerak.

Algoritma ini memiliki beberapa kelebihan yakni: (1) algoritma sangat mudah dipahami dan diimplementasikan, (2) perhitungan pergerakan sangat sederhana dengan kompleksitas O(1) per agen dan O(n) secara keseluruhan, (3) memberikan acuan dasar untuk perbandingan dengan algoritma yang lebih kompleks, (4) hasil simulasi dapat diprediksi dan direproduksi.

Tentu saja algoritma ini memiliki kekurangan yang cukup jelas: (1) agen tidak mempertimbangkan agen lain dalam perencanaan jalur, (2) agen dapat saling bertabrakan di area sempit, (3) agen tidak dapat mengantisipasi potensi tabrakan, (4) hanya dapat menyelesaikan kejadian tabrakan dengan mendorong satu sama lain.

Algoritma Ghost berfungsi sebagai *baseline benchmark* dengan menyediakan metrik pembanding di saat agen tidak memiliki algoritma apa pun dalam upaya menghindari tabrakan.

### *C. Metode Reaktif (Boids)*

Algoritma Boids, dikembangkan oleh Craig Reynolds pada tahun 1987, merupakan salah satu pendekatan klasik dalam simulasi perilaku kerumunan (*flocking behavior*). Nama "Boids" merupakan singkatan dari "bird-oids" atau objek yang menyerupai burung. Algoritma ini dibuat berdasarkan tiga aturan fundamental: *separation* (pemisahan), *alignment* (penyelarasan), dan *cohesion* (kohesi).

Dalam implementasi ini, fokus diberikan pada pendekatan yang disederhanakan dengan menggunakan hanya *separation rule* yang dikombinasikan dengan *goal-seeking behavior* untuk menciptakan perilaku navigasi yang efektif namun tetap mempertahankan karakteristik reaktif dari algoritma Boids.

Algoritma Boids menggunakan konsep *steering behaviors* yang dikembangkan oleh Craig Reynolds. Konsep ini memodelkan pergerakan agen sebagai kombinasi dari berbagai "kekuatan" (*forces*) yang mempengaruhi arah dan kecepatan pergerakan.

$$F_{total} = F_{separation} + F_{goal}$$

$F_{total}$ : Kekuatan total yang mempengaruhi agen

$F_{separation}$ : Kekuatan pemisahan dari agen lain

$F_{goal}$ : Kekuatan menuju tujuan

Implementasi ini hanya menggunakan *separation rule* dari tiga aturan klasik Boids dengan alasan: (1) *separation rule* adalah aturan yang paling penting untuk mencegah tabrakan antar agen, yang merupakan masalah utama dalam simulasi kerumunan, (2) kombinasi *separation* dengan *goal-seeking* menghasilkan perilaku yang realistis tanpa kompleksitas berlebihan dari *alignment* dan *cohesion*, (3) Dengan hanya dua perilaku utama, pengaturan parameter menjadi lebih mudah dan hasil lebih mudah diprediksi.

Untuk menghitung *separation force*, algoritma menggunakan formula matematika sebagai berikut:

$$F_{separation} = \sum_{i=1}^{n} \frac{w_i \times (P_{agent} - P_i)}{||P_{agent} - P_i||}$$

$n$ : jumlah agen dalam radius pemisahan

$w_i$ : bobot berdasarkan jarak ke agen ke-i

$P_{agent}$ : posisi agen saat ini

$P_i$ : posisi agen ke-i

Untuk menghitung goal force, algoritma menggunakan formula matematika sebagai berikut:

$$F_{goal} = ||P_{goal} - P_{agent}|| \times v_{max}$$

Kemudian, *total forces* akan dihitung dengan menggunakan metode *weighted sum*:

$$F_{total} = F_{separation} \times w_{separation} + F_{goal} \times w_{goal}$$

Algoritma Boids akan mengecek setiap agen selain dirinya sendiri satu per satu untuk mengetahui jaraknya dan menentukan gaya yang mempengaruhinya. Oleh karena itu, kompleksitas waktu dari algoritma ini adalah $O(n^2)$ dan memiliki kompleksitas ruang sebagai $O(n)$ karena menyimpan setiap agen.

Algoritma ini dapat dioptimisasi dengan sistem Divide and Conquer menggunakan struktur data Quadtree. QuadTree akan membagi ruang 2D menjadi empat kuadran secara rekursif. Setiap node dapat memiliki maksimal empat

anak, masing-masing merepresentasikan kuadran: NW, NE, SW, SE.

Optimisasi quadtree akan dimulai dengan satu segi empat besar yang mencakup keseluruhan tempat. Kemudian, berdasarkan banyaknya agen dalam suatu persegi, bagi area persegi tersebut menjadi empat bagian (NW, NE, SW, SE). Ulangi proses ini hingga suatu area persegi terlalu kecil atau banyak agen sudah di bawah suatu ambang batas.

Ketika suatu agen ingin mengecek agen lain yang berada di sekitarnya, cukup mengecek kuadran yang relevan pada quadtree. Pengecekan Boids berdasarkan suatu radius pencarian. Untuk menentukan area mana saja yang perlu dicek oleh suatu agen, algoritma akan membuat sebuah lingkaran dengan radius sebesar radius pencarian dan hanya periksa area quadtree yang bersentuhan dengan lingkaran ini.

Dengan menggunakan pendekatan quadtree, kompleksitas waktu akan menjadi $O(n \log n)$ dan kompleksitas ruang tetap sama. Metode optimisasi ini akan menjadi efisien jika $n^2 > n \log n + overhead$, dengan $n$ adalah banyaknya agen. Dalam konteks ini, efisiensi akan tercapai jika $n > 50 - 100$ agen.

Algoritma Boids modifikasi ini memiliki beberapa kelebihan sebagai berikut:

1. Menghasilkan pola pergerakan yang kompleks dari aturan sederhana
2. Parameter mudah disesuaikan dengan mengubah bobot dan radius
3. Agen menghindari tabrakan sambil tetap bergerak menuju tujuan
4. Konsep mudah dipahami dan diimplementasikan karena hanya menerapkan sebagian dari *boids rule*
5. Perilaku konsisten dan dapat diprediksi

Namun, ada juga beberapa kelemahan sebagai berikut:

1. $O(n^2)$ tidak scalable untuk simulasi besar kecuali jika menggunakan optimisasi quadtree
2. Tidak ada pathfinding sehingga tidak dapat mengatasi rintangan kompleks
3. Dapat terjebak dalam situasi deadlock
4. Jika menggunakan optimisasi quadtree, perlu penyesuaian parameter quadtree dan membuat quadtree setiap frame

*D. Metode Prediktif (RVO)*

Metode RVO (Reciprocal Velocity Obstacles) merupakan implementasi dari pendekatan prediktif dalam simulasi kerumunan yang dikembangkan berdasarkan konsep Velocity Obstacles. Algoritma yang mengimplementasi metode ini akan dinamakan "Scout" karena setiap agen bertindak seperti pengintai yang memprediksi masa depan untuk merencanakan jalur yang aman sebelum bergerak.

RVO berbeda secara fundamental dari algoritma reaktif seperti Boids. Jika Boids bereaksi terhadap situasi saat ini, RVO memprediksi kemungkinan tabrakan di masa depan dan merencanakan pergerakan untuk menghindarinya sebelum terjadi.

Velocity Obstacle (VO) adalah area dalam "ruang kecepatan" yang merepresentasikan semua kecepatan yang akan menyebabkan tabrakan dengan agen lain dalam periode waktu tertentu. Dengan kata lain, $VO_{A|B}$ adalah sekumpulan vektor kecepatan A terhadap B yang jika digunakan oleh A, akan membuat A menabrak B.

Untuk menentukan $VO_{A|B}$ perlu beberapa langkah. Langkah pertama adalah penentuan posisi dan kecepatan relatif antara A dan B.

$$\mathbf{v}_{BA} = \mathbf{v}_A - \mathbf{v}_B$$
$$\mathbf{p}_{BA} = \mathbf{p}_B - \mathbf{p}_A$$

Kemudian, untuk mempermudah penghitungan, kita anggap agen A sebagai suatu titik yang berada di pusat agen A. Karena agen A diperkecil menjadi titik, agen B perlu diperbesar sehingga radius agen B menjadi:

$$r = r_A + r_B$$

Penjumlahan ini juga disebut sebagai Minkowski Sum. Kemudian, kita perlu menentukan kerucut $VO_{rel}$ yakni vektor-vektor dari titik A yang akan menyentuh area agen B. Arah sumbu kerucut adalah arah $\mathbf{p}_{BA}$ dan besar sudut kerucut adalah:

$$\theta = sin^{-1}(\frac{r}{||\mathbf{p}_{BA}||})$$

Secara formal,

$$VO_{A|B} = \{\mathbf{v} | \exists t > 0, (\mathbf{v} \cdot \mathbf{p}_{BA}) > cos(\theta) \cdot ||\mathbf{v}||\}$$

Terakhir, transformasi $VO_{A|B}$ ke dalam ruang kecepatan A dengan:

$$VO_{A|B} = \mathbf{v}_B + VO_{rel}$$

Jika $\mathbf{v}_A$ berada dalam himpunan $VO_{A|B}$, agen A akan menabrak agen B.

Yang membedakan antara VO dengan RVO adalah perhitungan dalam tranformasi VO relatif. Pada RVO transformasi menjadi

$$VO_{A|B} = \frac{\mathbf{v}_A + \mathbf{v}_B}{2} + VO_{rel}$$

Dalam konteks VO, agen A dianggap menentukan VO dan agen B bergerak tanpa memperhitungkan apa pun. Metode RVO menganggap bahwa A dan B akan sama-sama menghitung VO sehingga terdapat perbedaan dalam perhitungan transformasi. Anggap pada suatu lorong terdapat

agen A dan agen B. mereka berhadap-hapadan. Untuk menghindari tabrakan, agen A bergerak ke kiri. Namun, agen B juga ikut bergerak ke kiri untuk menghindari tabrakan. Terjadilah suatu situasi *livelock*. RVO mengatasi masalah ini dengan menganggap bahwa kedua agen membuat VO sehingga pergeseran terhadap VO relatif menjadi rata-rata vektor kecepatan A dan B. Kompleksitas waktu untuk metode ini adalah $O(n^2)$ karena satu agen harus mengecek agen lainnya. Algoritma ini juga memiliki kompleksitas ruang sebagai $O(n)$ karena menyimpan setiap agen.

Optimisasi QuadTree juga dapat diterapkan dalam implementasi Algoritma Scout ini. Dengan pendekatan yang hampir sama seperti boids, kompleksitas waktu menjadi $O(n \log n)$ dan juga menjadi efisien ketika $n^2 > n \log n + overhead$.

Algoritma RVO ini memiliki beberapa kelebihan sebagai berikut:

1. Algoritma ini memiliki pergerakan yang lebih halus dan natural
2. Pengambilan keputusan dianggap bersama (*reciprocal*) sehingga mencegah *oscillation behavior* dan mengurangi *deadlock/livelock*
3. Algoritma ini meminimalisasi deviasi dari jalur lurus menuju target demi menghindari tabrakan.

Namun, ada juga beberapa kelemahan sebagai berikut:

1. Kompleksitas komputasi tinggi, dengan *sampling* 360 vektor kecepatan per agen dan perhitungan geometri yang kompleks
2. Jika tidak ada vektor kecepatan yang aman, akan terjadi *deadlock* di mana agen memiliki kecepatan nol.

### *E. Pemodelan Collision Resolution*

Dalam simulasi tabrakan, model harus mampu merepresentasikan bagaimana agen-agen saling bertabrakan satu sama lain. Namun, sering kali muncul masalah di mana agen-agen justru saling menembus ketika tabrakan tidak dapat dihindari. Untuk mengatasi hal ini dan memastikan bahwa agen tidak saling menembus, digunakan algoritma penyelesaian tabrakan (*collision resolution*) yang sederhana. Algoritma ini bertugas mensimulasikan efek fisik dari tabrakan dengan cara memposisikan ulang agen-agen yang terlibat agar mereka tetap berada di posisi yang logis dan tidak saling bertumpang tindih.

Syarat dua agen bertabrakan adalah jarak kedua agen tersebut lebih kecil dari jumlah radius kedua agen dan jarak kedua agen lebih besar dari nol (agar menghindari pembagian dengan nol). Kemudian, setiap langkah simulasi (*frame*), algoritma akan menghitung seberapa besar tumpang tindih antara dua agen. Lalu, algoritma akan menghitung vektor pemisahan yakni arah pemisahan dari kedua agen. Setelah itu, kedua agen akan digerakkan searah dengan vektor pemisahan dan sebesar setengah dari jarak tumpang tindih.

Untuk metode tanpa QuadTree, kompleksitas waktu algoritma *collision resolution* adalah $O(n^2)$. Jika menggunakan optimisasi QuadTree, kompleksitas waktu menjadi $O(n \log n)$.

## III. Hasil Eksperimen dan Analisis

Metrik untuk eksperimen ini adalah sebagai berikut:

1. Durasi simulasi
2. Total agen
3. Rata-rata, min, max, dan standar deviasi FPS
4. Rata-rata, min, max, dan standar deviasi waktu komputasi
5. Rata-rata jarak ditempuh
6. Banyak tabrakan

Eksperimen dilakukan dengan pola agen melingkar dengan target berada pada posisi agen yang berhadapan.

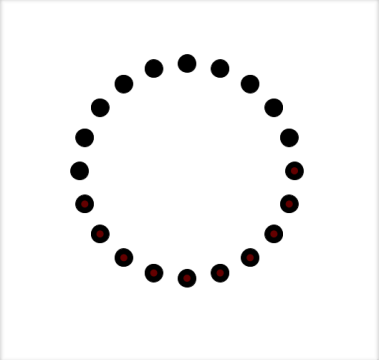

Fig. 1. Visualisasi posisi agen dalam skenario satu.

Dilakukan 5 kali percobaan dengan banyak agen 100 untuk tiap algoritma.

Metode Naif

| | |
|---|---|
| Durasi Simulasi | 6.62 detik |
| Total agen | 100 agen |
| Rata-rata<br>Min<br>Max<br>Standar deviasi<br>FPS | 73.54<br>3.29<br>3008.37<br>1104.78 |
| Rata-rata<br>Min<br>Max<br>Standar deviasi<br>Waktu komputasi | 0.368 ms/frame<br>0.068 ms<br>15.983 ms<br>1.154 ms |
| Rata-rata jarak ditempuh | 3012.16 units |
| Banyak tabrakan | 13456 |

Metode Boid QuadTree

| | |
|---|---|
| Durasi Simulasi | 17.84 detik |
| Total agen | 100 agen |
| Rata-rata<br>Min<br>Max<br>Standar deviasi<br>FPS | 63.97<br>10.18<br>3786.73<br>960.15 |
| Rata-rata<br>Min<br>Max<br>Standar deviasi<br>Waktu komputasi | 0.292 ms/frame<br>0.069 ms<br>8.033 ms<br>0.618 ms |
| Rata-rata jarak ditempuh | 7178.29 units |
| Banyak tabrakan | 1926 |

Metode RVO QuadTree

| | |
|---|---|
| Durasi Simulasi | 11.48 detik |
| Total agen | 100 agen |
| Rata-rata<br>Min<br>Max<br>Standar deviasi<br>FPS | 77.87<br>8.94<br>3675.05<br>1249.55 |
| Rata-rata<br>Min<br>Max<br>Standar deviasi<br>Waktu komputasi | 2.859 ms/frame<br>0.067 ms<br>17.665 ms<br>2.939 ms |
| Rata-rata jarak ditempuh | 3259.83 units |
| Banyak tabrakan | 126 |

Metode Naif merupakan pendekatan dasar tanpa optimasi dalam pencarian tetangga agen. Hasil eksperimen menunjukkan bahwa metode ini memiliki durasi simulasi tercepat, yaitu rata-rata 6,62 detik. Hal ini sejalan dengan FPS rata-rata tertinggi kedua di antara ketiga metode, yaitu 73,54 FPS. Namun, perlu dicatat bahwa walaupun performa waktu cukup cepat, metode ini menghasilkan jumlah tabrakan yang sangat tinggi, yaitu sebanyak 13.456 tabrakan.

Waktu komputasi per frame metode ini rata-rata hanya 0,368 ms, menunjukkan efisiensi dalam proses per frame. Namun, tingginya jumlah tabrakan mengindikasikan bahwa metode ini kurang akurat dalam menghindari tabrakan, akibat dari ketiadaan strategi prediksi dan penghindaran konflik.

Rata-rata jarak tempuh agen dalam metode ini adalah 3.012,16 satuan unit. Hal ini dapat mengindikasikan bahwa pergerakan agen cukup aktif, namun bisa juga disebabkan oleh manuver yang tidak efisien akibat seringnya tabrakan.

Metode kedua menggunakan algoritma Boid yang digabungkan dengan struktur data QuadTree untuk mengoptimalkan pencarian tetangga. Durasi simulasi meningkat menjadi 17,84 detik, namun diimbangi dengan penurunan signifikan dalam jumlah tabrakan, yakni hanya 1.926 tabrakan – turun sekitar 85% dibanding metode Naif.

FPS rata-rata menurun menjadi 63,97, yang masih dalam kategori dapat diterima untuk simulasi real-time, terutama mengingat peningkatan akurasi. Waktu komputasi per frame mengalami sedikit peningkatan efisiensi dibanding metode Naif, yakni 0,292 ms/frame. Hal ini menunjukkan bahwa penggunaan QuadTree memang membantu mempercepat proses pencarian tetangga, walau durasi simulasi lebih panjang kemungkinan akibat kompleksitas interaksi antaragen yang lebih realistis.

Yang paling menonjol dari metode ini adalah rata-rata jarak tempuh agen yang mencapai 7.178,29 unit – lebih dari dua kali lipat dibanding metode Naif. Ini menunjukkan bahwa agen dapat bergerak lebih leluasa dan efisien dalam ruang simulasi, berkat penghindaran tabrakan yang lebih baik.

Metode ketiga menggunakan pendekatan Reciprocal Velocity Obstacles (RVO) dikombinasikan dengan QuadTree. Hasilnya cukup menarik, dengan durasi simulasi selama 11,48 detik dan FPS tertinggi di antara ketiganya, yaitu 77,87. Artinya, metode ini mampu menjalankan simulasi dengan sangat lancar. Akan tetapi, waktu komputasi per frame meningkat signifikan menjadi rata-rata 2,859 ms, menandakan bahwa proses pengambilan keputusan untuk setiap agen jauh lebih kompleks dibanding dua metode sebelumnya.

Dari segi jumlah tabrakan, metode ini unggul jauh dibanding dua lainnya dengan hanya 126 tabrakan – menjadikannya metode paling efektif dalam menghindari tabrakan. Rata-rata jarak tempuh agen adalah 3.259,83 unit, sedikit lebih tinggi dari metode Naif, namun jauh lebih rendah dibanding metode Boid. Hal ini bisa diartikan bahwa pergerakan agen pada metode RVO cenderung lebih hati-hati dan efisien, menghindari jalur yang berisiko tinggi terhadap konflik.

## IV. Kesimpulan dan Saran

### *A. Kesimpulan*

Penelitian ini bertujuan untuk mengevaluasi dan membandingkan performa tiga pendekatan algoritma dalam simulasi kerumunan, yaitu metode Naif, Boid, dan RVO, dalam konteks penghindaran tabrakan. Masing-masing algoritma diuji dengan jumlah agen yang sama dan skenario simulasi identik, serta diukur dengan berbagai metrik seperti FPS, waktu komputasi, jarak tempuh, dan jumlah tabrakan.

Berdasarkan hasil eksperimen dan analisis yang telah dilakukan, dapat disimpulkan beberapa hal berikut:

1. Metode Naif memberikan performa simulasi tercepat (rata-rata durasi 6,62 detik) dengan waktu komputasi yang ringan (0,368 ms/frame), namun menghasilkan jumlah tabrakan yang sangat tinggi (13.456 tabrakan).
2. Metode Boid QuadTree menunjukkan keseimbangan yang cukup baik antara efisiensi dan efektivitas. Meskipun memiliki durasi simulasi lebih panjang (17,84 detik), algoritma ini mampu mengurangi tabrakan hingga 85% dibanding metode Naif (hanya 1.926 tabrakan). Selain itu, rata-rata jarak tempuh agen tertinggi (7.178,29 unit) menunjukkan bahwa agen dapat bergerak lebih bebas dan realistis di lingkungan simulasi.
3. Metode RVO dengan QuadTree menjadi yang paling efektif dalam menghindari tabrakan dengan hanya 126 tabrakan, angka yang secara signifikan lebih rendah dibanding dua metode lainnya. FPS tertinggi (77,87) menandakan kelancaran simulasi, meskipun dengan waktu komputasi per frame yang jauh lebih tinggi (2,859 ms/frame). RVO menampilkan perilaku agen yang lebih hati-hati, efisien, dan prediktif.
4. Penerapan optimisasi menggunakan struktur data QuadTree terbukti memberikan peningkatan signifikan dalam performa pencarian tetangga, terutama pada algoritma Boid dan RVO yang memiliki kompleksitas tinggi. Hal ini mengindikasikan bahwa integrasi strategi divide and conquer pada skenario kerumunan berskala besar merupakan pendekatan yang efektif.

## *B. Saran*

Berdasarkan temuan dan keterbatasan dari penelitian ini, beberapa saran untuk pengembangan lebih lanjut antara lain:

1. Penelitian selanjutnya dapat mengeksplorasi integrasi sistem pathfinding seperti A* atau Dijkstra untuk memungkinkan agen menghindari rintangan tetap (seperti dinding atau objek statis), sehingga lingkungan simulasi lebih realistis.
2. Uji coba algoritma dengan jumlah agen yang lebih besar (> 1000 agen) perlu dilakukan untuk mengetahui sejauh mana performa optimisasi QuadTree tetap efektif, dan bagaimana kestabilan algoritma dipertahankan dalam skala besar.
3. Penggabungan model kepanikan, prioritas individu, atau emosi sosial dapat meningkatkan realisme simulasi, khususnya dalam skenario darurat seperti evakuasi massal.
4. Algoritma *collision resolution* yang lebih kompleks, misalnya berbasis fisika atau impuls, dapat digunakan untuk memperbaiki manajemen interaksi saat agen benar-benar bertabrakan dan memperbaiki kelemahan dari model resolusi sederhana yang digunakan saat ini.
5. Selain QuadTree, struktur data spasial lain seperti k-d tree, grid hashing, atau even bounding volume hierarchy (BVH) dapat dibandingkan untuk mengetahui struktur yang paling optimal dalam konteks simulasi agen dinamis.

## REFERENCES


[1] Sharadhi A.K (2025). Scouring Through the Crowd Simulation Dynamics in Urban Environments. International Journal of Innovative Science and Research Technology, 10(3), 1439-1459. https://doi.org/10.38124/ijisrt/25mar1176

[2] Ali, S., Nishino, K., Manocha, D., & Shah, M. (2013). Modeling, Simulation and Visual Analysis of Crowds: A Multidisciplinary Perspective. The International Series in Video Computing, 1–19. doi:10.1007/978-1-4614-8483-7_1

[3] Y. Yang et al., “A review on crowd simulation and modeling,” UC Davis Mathematics. [Online]. Available: https://www.math.ucdavis.edu/~saito/data/PSO-ACO/yang-etal_review-crowd-simulation.pdf. [Accessed: Jun. 22, 2025].


## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Juni 2025

Rafael Marchel Darma Wijaya 13523146